

**METHOD AND APPARATUS FOR TRACING PACKETHASH-BASED SYSTEMS
AND METHODS FOR DETECTING, PREVENTING, AND TRACING NETWORK
WORMS AND VIRUSES**

Field of the Invention

The present invention relates generally to ~~the field of~~ network security and, more specifically~~particularl~~y, to ~~using low overhead~~systems and methods for ~~identifying~~detecting ~~and/or preventing the intrusion location~~ transmission of a packet in ~~malicious packets, such as worms and viruses, and tracing their paths through~~ a network.

Description of Related Art

Availability of low cost computers, high speed networking products, and readily available network connections has helped fuel the proliferation of the Internet. This proliferation has caused the Internet to become an essential tool for both the business community and private individuals. Dependence on the Internet arises, in part, because the Internet makes it possible for multitudes of users to access vast amounts of information and perform remote transactions expeditiously and efficiently. Along with the rapid growth of the Internet have come problems caused by malicious individuals or pranksters launching attacks from within the network. As the size of the Internet continues to grow, so does the threat posed by these individuals.

The ever-increasing number of computers, routers, and connections making up the Internet increases the number of vulnerability points from which these malicious individuals can launch attacks. These attacks can be focused on the Internet as a whole or on specific devices, such as hosts or computers, connected to the network. In fact, each router, switch, or computer

connected to the Internet may be a potential entry point from which a malicious individual can launch an attack while remaining largely undetected. Attacks carried out on the Internet often consist of malicious packets being injected into the network. Malicious packets can be injected directly into the network by a computer, or a device attached to the network, such as a router or switch. ~~Such a computer or device,~~ can be compromised and configured to place malicious packets onto the network.

~~{001} ——— The most publicized forms of network attacks often involve placing thousands or millions of packets onto the network using a practice known as *flooding*. The flood of packets can be targeted to a specific device on the network, for example a corporate web site, thus causing the device to become overwhelmed and shutdown. Alternatively, an attack may be designed to clog the links, or connection points, between network components. Network attacks can be further enhanced using a practice known as *spoofing*. Spoofing involves associating bogus Internet Protocol (IP) addresses with transmitted packets, thus making the packets' origins impossible to determine based upon looking only at a received packet. Spoofing can be further enhanced using a technique referred to as *transformation*. When a packet is transformed, it undergoes a process that changes the original packet into a new packet, as, for example, would happen during tunneling or network address translation (NAT). Locating the origin of a network attack is further complicated because *coordinated attacks* can be employed. In a coordinated attack, multiple network devices are compromised and then used to launch a *distributed attack*. A distributed attack is one that is launched essentially simultaneously from several locations within the network.~~

~~{002} ——— Network attacks can also be launched using a single packet. While single packet attacks are not as well publicized as multi-packet attacks, they are becoming more common and they are capable of inflicting significant damage to vulnerable networks. At present, it is extremely difficult to detect single packet attacks in a timely manner using known methods of intrusion detection, which exacerbates the challenge in dealing with them. As a result, network data, currently, must be analyzed after the fact to determine if a single packet attack was the source of disruption. Any tracing of the single packet to its origins, in accordance with prior art techniques, must also take place after the attacking packet traversed the network.~~

[003] ——— Much of the difficulty in identifying the origin of an attack arises because the Internet employs a stateless routing infrastructure, in that it is one in which routing is based solely on destination addresses. Although source IP addresses may be transmitted with data, they are easy to forge, and as a result they are untrustworthy. A forged source address may bear no similarity to the actual source address from which the packet came. As a result, most prior art techniques and devices for preventing network attacks attempt to stop delivery of malicious packets at the ultimate destination device rather than attempting to locate their origin. Such origin is referred to as an *entry point*, also referred to as an *ingress point* or *intrusion location*, onto the network. Failing to identify the source address of malicious packets inhibits preventing further attacks, and such failure makes identification of the actual perpetrator difficult.

Figure 1

[004] ——— Fig. 1 provides an example of a network employing prior art devices to thwart malicious packets. Two prior art autonomous systems are shown, PAS1 and PAS2, respectively, connected to the Internet, or public network (PN1) shown comprised of routers R2-R6. An autonomous system (AS) is a network domain in which all routers in the AS can exchange routing tables. Often the AS may be a local area network (LAN) such as one found at a university, municipality, large corporation, or Internet Service Provider (ISP). An AS may further be comprised of computers, or hosts, connected to the AS such as H1-H3 for PAS1 or H4-H5 for PAS2, respectively. An AS is normally connected to the public network by one or more border routers, here R1 (for PAS1) or a firewall F1 (for PAS2) incorporating router functionality.

[005] ——— Border routers contain routing tables for other routers within the AS and for routers within the public network that are connected to the AS by a link, i.e. a communicative connection. In Fig. 1, R1 is a border router for PAS1 and it connects to the Internet using representative link L1. Routing tables act as road maps for routers on the network, in that they are used to ensure that network traffic is forwarded through the appropriate links in route to a desired destination address.

[006] ——— Firewalls are typically installed between a local area network (LAN), or intranet, and the Internet, or public network. Firewalls act as gatekeepers for an AS in that they allow

certain packets in while excluding other packets. Firewalls may be implemented in routers or servers connected between an AS and the Internet, or they may function as standalone devices. Rule sets are used by firewalls to determine which packets will be allowed into their respective AS and which packets will be discarded. Since rules determine which packets get through the firewalls, only packets known to be problematic can be stopped. Therefore, rule sets must be updated on a regular basis to provide protection against new threat characteristics.

{007}—Additional protection for an AS may be obtained by supplementing border routers and firewalls with intrusion detection systems (IDSs). IDSs also use rule-based algorithms to determine if a given pattern of network traffic is abnormal. The general premise used by an IDS is that malicious network traffic will have a different pattern from normal, or legitimate, network traffic. Using a rule set, an IDS monitors inbound traffic to an AS. When a suspicious pattern or event is detected, the IDS may take remedial action, or it can instruct a border router or firewall to modify operation to address the malicious traffic pattern. For example, remedial actions may include disabling the link carrying malicious traffic, discarding packets coming from a particular source address, or discarding packets addressed to a particular destination. In Fig. 1, IDS1 is used to protect PAS1 and IDS2 is used in conjunction with F1 to protect PAS2.

{008}—Although border routers, firewalls, and IDSs can be used to help prevent known packets from entering an AS, they are not well equipped for stopping unknown packets because they rely on rule-based look up tables containing signatures of known threats. In addition, border routers, firewalls, and IDSs generally are not well equipped for identifying the origin, or ingress location, of malicious packets, particularly when spoofing is employed. Even when spoofing is not used, the above-noted devices may not be able to determine the ingress point for packets because packets often traverse many Internet links and devices, such as routers, bridges, and switches, before arriving at an AS. Reliably tracing the path of a packet often requires information about each link traversed by a packet. To obtain this information, routing data must remain with the packet or, alternatively, each router, or device, on the path must store information about, or a copy of, each packet traversing a network. With high-speed routers passing gigabits of data per second, storing full copies of packets is not practical.

~~{009} ——— What has been needed and what has not been available is a method for identifying the origin of malicious packets that can be implemented in an AS on the Internet and which addresses all shortcomings of prior art protection techniques. Embodiments of the present invention offer welcome solutions to these prior art protection problems.~~

~~SUMMARY OF THE INVENTION~~

~~{0010} ——— Embodiments of the present invention employ apparatus, system, computer program product and/or method for identifying an intrusion point of a malicious or target packet into a network. More specifically, in a network including multiple hosts and multiple routers for facilitating transmission of packets on a network, a system, for example, is employed for determining the point of entry of a malicious packet. An intrusion detection system detects the entry of a malicious packet in the network. A source path isolation server responsive to the intrusion detection system isolates the malicious packet and thereby determines the point of entry of the malicious packet. In a further embodiment of the system, the source path isolation server includes a means for generating a query message containing information about the malicious packet and a means for forwarding the query message to some of the routers located one hop away. In still a further embodiment of the system, certain of the routers include means for generating a hash value of the identification information about the malicious packet, a means for establishing a bit map of hash values representative of packets having passed through the respective router, and a means for comparing the hash value of the identification information to the hash values of packets having passes through the respective router.~~

~~{0011} ——— In a further aspect of the invention, in a network carrying a plurality of packets where at least one of the packets is a target packet, the network includes at least one network component, a detection device and a server, a technique for determining the point of entry of a target packet into the network. The target packet is received from the detection device at the server. A query message is sent to a first one of the network components where the query message identifies the target packet. A reply containing information about the target packet from the first network component is received. The reply is processed to extract information~~

contained therein. And, the information is used in a manner that allows the entry point of the target packet to ultimately be determined.

{0012} ——— In yet a further aspect of the invention, in a network carrying a plurality of packets, a computer readable data signal is embodied in a transmission medium used to identify an intrusion location of a target packet. The network includes a server and a network component having a memory storing representations of the plurality of packets, namely the data signal. A header portion includes an address of the network component. And, a body portion includes at least a portion of the target packet, the body portion being compared to corresponding representations where a match between a portion of the target packet and one of the representations indicates that the network component encountered the target packet.

{0013} ——— In still a further aspect of the invention, in a network carrying a plurality of packets, the network includes a network component having a memory storing first information about a subset of the plurality of packets having passed through the network component. The network component further includes a processor for computing a first hash value of a target packet and a second hash value of a member of the subset of the plurality of packets. The memory also stores second information about an intrusion location of the target packet in the network. A data structure stored in the memory includes information resident in a database used by a source-path isolation program for determining the intrusion location with the data structure. A network component identification attribute corresponds to a location of the network component. A target packet attribute uniquely identifies the target packet. And, a reply packet attribute associated with at least one of the members and being associated with the network component identification attribute identifies the origin of the reply packet with the reply packet indicating that the member was encountered if the first hash value matches the second hash value.

{0014} ——— It is advantageous to employ embodiments of the present invention to protect data networks. A further advantage of the invention is the elimination of problems caused by undetected malicious packets in a network. A still further advantage of the invention is that it detects malicious packets without requiring special purpose network equipment. Furthermore, the present invention communicates information about malicious packets to other network

devices thus enhancing network security. Another advantage of the invention is that it efficiently uses stored information about packets to facilitate detecting malicious packets.

{0015} — It is thus a general object of the present invention to provide improved packet networks.

{0016} — It is another object of the present invention to eliminate problems caused by malicious packets in a network.

{0017} — It is a further object of the present invention to identify malicious packets to facilitate identifying their intrusion locations into the network.

{0018} — It is a further object of the present invention to quickly identify ingress points of malicious packets when distributed attacks are launched against a network.

{0019} — It is yet a further object of the present invention to efficiently use stored information about packets traversing a link in a network.

{0020} — Further objects and advantages of the present invention will become more apparent after reference to the detailed description of exemplary embodiments thereof taken in conjunction with the accompanying drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

{0021} — Fig. 1 is a block diagram of a prior art network comprising autonomous systems;

{0022} — Fig. 2 is a block diagram of an exemplary embodiment of the present invention operating in conjunction with an Internet network;

{0023} — Fig. 3 is a schematic diagram of an autonomous system coupled to a plurality of external networks;

{0024} — Fig. 4 is a flowchart illustrating an exemplary method for use with a source path isolation server;

{0025} ——— Fig. 5 is a schematic diagram of an exemplary data structure for storing information in a source-path isolation server for use in performing source-path isolation techniques; and

{0026} ——— Fig. 6 is a block diagram of a general-purpose computer-configurable for practicing exemplary embodiments of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

Figure 2

{0027} ——— A preferred embodiment uses a server and one or more specially configured network components, or devices, such as a router, within an autonomous system (AS) to determine the ingress point, or location, for a malicious packet (MPI). Fig. 2 illustrates an embodiment that may be used with an Internet Protocol network. More particularly, Fig. 2 is broken into three general areas enclosed within borders with communication media, such as links, carrying data traffic across the network, connecting the general areas. Links serve as a transmission media for data and signals on the network and may be comprised of wire, optical fiber, radio-frequency (RF) transponders, or the like.

{0028} ——— The rightmost portion of Fig. 2 denotes an AS, shown as AS1, enhanced by the addition of a source path isolation server (SS1) and network components, here routers, modified to work as source path isolation routers (SRs), denoted by SR14-17, respectively. Also included within AS1 is a detection device, here an intrusion detection system (IDS) denoted as IDS1, and host computers H1-H3. IDS1 may take the form of a commercially available IDS, or alternatively it may be developed specifically for participating in source path isolation systems and methods. IDSs and firewalls are well known in the art and will not be described in detail herein. An informative source of information on IDS and firewall functionality that may be used with the disclosed embodiments can be found in *Firewalls and Internet Security: Repelling the Wily Hacker*, by William R. Cheswick and Steven M. Bellowin, Addison-Wesley (1994).

{0029} ——— SS1 may be comprised of a general-purpose computer, or server, operatively coupled to the network of AS1 and executing machine-readable code enabling it to perform

source path isolation in conjunction with SR14-17 and IDS1. While SS1 and IDS1 are shown as separate devices in Fig. 2, it is noted that they can be combined into a single unit performing both intrusion detection and source path isolation. SR14-17 may be comprised of commercially available routers, or similar devices such as switches, bridges or the like, employing software and hardware enabling them to participate in source path isolation.

{0030} — The central portion of Fig. 2 represents the public network, shown as PN1, carrying traffic between the autonomous systems, namely IAS1, and AS1, AS2 and AS3. PN1 comprises routers R2-R6, links operatively coupling the routers making up PN1, and links attaching to ASs coupled to PN1. PN1 may also comprise computers external to an AS (not shown). In the foregoing discussion, routers that have not been modified to act as source path isolation routers (SRs) are denoted as Rx, such as those located in PN1, where x is a number such as 2, 3, 4, etc..

{0031} — The lower portion of Fig. 2 includes other autonomous systems, AS2 and AS3 that may be operatively connected to PN1. AS2 and AS3 may employ source path isolation apparatus and methods, or alternatively, they may be prior art autonomous systems (PAS).

{0032} — The leftmost portion of Fig. 2 shows an autonomous system (IAS1) used by an intruder to launch an attack on AS1. IAS1 contains an IDS, shown as IDS2, operatively coupled to three host computers H4, H5 and H1 using links. In Fig. 2, H1 has been configured such that it places a malicious packet (MP1) onto IAS1 for transmission to AS1 via PN1. While Fig. 2 illustrates a computer configured to place MP1 onto the network, routers, switches, gateways and other hardware capable of placing machine-readable data onto a network may be used in place of or in conjunction with such computer. When a device has been configured to inject an MP1 onto a network, it is referred to as an *intruder or intruding device*.

{0033} — To launch an attack, an intruder generates malicious data traffic and places it onto a link for transmission to one or more destination devices having respective destination addresses. In Fig. 2, the heavy lines are used to indicate the path taken by MP1, namely H1 to IDS2, IDS2-R6, R6-R3, R3-R2, R2-SR15, SR15-SR16, and SR16-IDS1 (where hyphenation implies operative coupling between network components). The thick dashed link from IDS1-H3 denotes the intended path to the targeted device H3.

[0034] — Detection and source path isolation of MPI may be accomplished as follows. Detection device, here IDS1, identifies MPI using known methods. After detecting MPI, IDS1 generates a notification packet, or triggering event, and sends it to SS1 thus notifying SS1 that a malicious packet has been detected within AS1. The notification packet may include MPI or portions thereof along with other information useful for SS1 to begin source path isolation. Examples of information that may be sent from IDS1 to SS1 along with MPI are time of arrival, encapsulation information, link information, and the like. When MPI (or fraction thereof) has been identified and forwarded to SS1 it is referred to as a target packet (TP1) because it becomes the target of the source path isolation method further described herein.

[0035] — SS1 may then generate a query message (QM1) containing TP1, a portion thereof, or a representation of TP1 such as a hash value. After generating QM1 containing identification information about TP1, SS1 sends it to some, or all, participating routers. Accordingly, SS1 may send QM1 to participating routers located one hop away; however the disclosed invention is not limited to single hops. For example, SR16 is one hop away from SS1, whereas SR14, SR15 and SR17 are two hops away from SS1 and one hop away from SR16, respectively. When SR16 receives QM1 from SS1, SR16 determines if TP1 has been seen. This determination is made by comparing TP1 with a database containing signatures of other characteristics representative of packets having passed through SR16. Typically, SR16 is considered to have observed, or encountered, a packet when the packet is passed from one of its input ports to one of its output ports such as would be done when SR16 forwards during normal operation within a network.

[0036] — To determine if a packet has been observed, SR16 first stores a representation of each packet it forwards. Then SR16 compares the stored representation to the information about TP1 contained in QM1. Typically, a representation of a packet passed through SR16 will not be a copy of the entire packet, but rather it will be comprised of a portion of the packet or some unique value representative of the packet. Since modern routers can pass gigabits of data per second, storing complete packets is not practical because memories become prohibitively large. In contrast, storing a value representative of the contents of a packet uses memory in a more efficient manner. By way of example, if incoming packets range in size from 256 bits to 1000 bits, a fixed width number may be computed across the bits making up a packet in a manner that

allows the entire packet to be uniquely identified. A hash value, or hash digest, is an example of such a fixed-width number. To further illustrate the use of representations, if a 32-bit hash digest is computed across each packet, then the digest may be stored in memory or, alternatively, the digest may be used as an index, or address, into memory. Using the digest, or an index derived therefrom, results in efficient use of memory while still allowing identification of each packet passing through a router. The disclosed invention works with any storage scheme that saves information about each packet in a space efficient fashion, that can definitively determine if a packet has not been observed, and that will respond positively (i.e. in a predictable way) when a packet has been observed. Although the invention works with virtually any technique for deriving representations of packets, for brevity, the remaining discussion will use hash digests as exemplary representations of packets having passed through a participating router.

{0037} ——— Returning to the discussion of Fig. 2, if SR16 has not observed TP1, it may so inform SS1. But if SR16 has a hash matching TP1, it may send a response to SS1 indicating that the packet was observed by, or at, SR16. In addition, SR16 may forward QM1 to adjacent routers 1 hop away. In Fig. 2, SR16 sends QM1 to SR14, SR15 and SR17. Then, SR14, 15 and 17 determine if they have seen TP1 and notify SS1 accordingly. In this fashion, the query message/reply process is forwarded to virtually all SRs within an AS on a hop-by-hop basis.

{0038} ——— In Fig. 2, routers SR14, SR15 and SR17 are border routers for AS1, namely they are the routers that contain routing tables for routers outside AS1. If routers external to AS1 have not been configured to operate as SRs, then the query message/reply process stops at SR14-17; however, if the public network routers are configured to act as SRs then the query message/reply process may continue until the SR closest to the ingress point of TP1 is reached. When the SR closest to the ingress point is found, it can be instructed to disconnect the link used by the intruder or it can be instructed to drop packets originating from the intruder's Internet Protocol (IP) address on a particular link, or based on other identifying information.

{0039} ——— Still referring to Fig. 2 and the route taken by MP1, if the routers making up PN1 are not participating as SRs, then SR15 would be instructed to exclude TPs. SR15 excludes a TP, present at an input port, by preventing it from passing to an output port. In contrast, if the

routers making up PN1 were participating as SRs then R6 could be instructed to exclude TPs present at its input port.

[0040] — The process used to perform source path isolation in Fig. 2 is referred to as an inward-out technique. After being triggered by an IDS, an inward-out technique begins its queries from a generally central portion of an AS. The inward-out technique then employs QMs that hop outward from the central portion of the AS toward the border routers comprised therein.

Figure 3

[0041] — Fig. 3 illustrates an autonomous system (AS), 300, employing border routers denoted generally as B connected to external networks EN1-EN7, other routers within 300 connected to the border routers generally denoted as A, and a source path isolation server denoted as SS. AS 300 may also include additional routers (not shown) located between SS and border routers B. An inward-out solution begins with SS at the center of Fig. 3 and works outward one hop at a time until the border routers, B, are reached. For Fig. 3, the routers labeled A are queried on the first hop and the border routers, B, are queried on a second, or subsequent, hop. Since the locations of border routers are known within AS 300, an outward-in solution may also be employed. With an outward-in solution, SS first queries the border routers, B, and they in turn query the routers labeled A. As can be seen from Fig. 3, an outward-in solution gets progressively closer to the center of AS 300. The disclosed technique can be used on networks containing virtually any number of participating routers. While inward-out and outward-in techniques have been herein described, the disclosed techniques are not limited to any particular types of solution or localization algorithms. Furthermore, SS may send queries to participating routers located virtually anywhere in the network so that many types of source path isolation techniques can be employed. Thus it can be seen that the disclosed technique is very scalable and flexible.

[0042] — Further detail of the operation of a source path isolation server (SS) and a source path isolation router (SR) are provided hereinbelow.

Figure 4

EXEMPLARY METHOD FOR SOURCE PATH ISOLATION SERVER

{0043} ——— Fig. 4 illustrates an exemplary method for accomplishing source path isolation. The method begins when SS1 receives TP1 from IDS1 operating within AS1 (step 402).

{0044} ——— After receiving TP1, SS1 may generate QM1 comprising TP1 and any additional information desirable for facilitating communication with participating routers (SRs) (step 404). Examples of additional information that may be included in QM1 are, but are not limited to, destination addresses for participating routers, passwords required for querying a router, encryption keying information, time to live (TTL) fields, a hash digest of TP1, information for reconfiguring routers, and the like. SS1 may then send QM1 to SRs located at least one hop away (step 406). SR may then process QM1 by hashing TP1 contained therein and comparing the resulting value to hash values stored in local memory, where the stored hash values identify packets having previously passed through SR.

{0045} ——— After processing QM1, an SR may send a reply to SS1 (step 408). The response may indicate that a queried router has seen TP1, or alternatively, that it has not (step 410). It is important to observe that the two answers are not equal in their degree of certainty. If SR does not have a hash matching TP1, SR has definitively not seen TP1. However, if SR has a matching hash, then SR has seen TP1 or a packet that has the same hash as TP1. When two different packets, having different contents, hash to the same value it is referred to as a *hash collision*.

{0046} ——— If a queried SR has seen TP1, a reply and identification (ID) information for the respective SR is associated as active path data (step 414). Alternatively, if an SR has not seen TP1, the reply is associated as inactive path data (step 412). Replies received from queried SRs are used to build a source path trace of possible paths taken by TP1 through the network using known methods (step 416). SS1 may then attempt to identify the ingress point for TP1 (step 418). If SS1 is unable to determine the ingress point of TP1, subsequent responses from participating routers located an additional hop away are processed by executing steps 408–418 again (step 424).

{0047} ——— Examples of source path tracing techniques that may be employed with embodiments disclosed herein are, but are not limited to, a breadth-first search or a depth-first search. In a breadth-first search, all SRs in an area are queried to determine which SRs may have observed a target packet. Then, one or more graphs, containing nodes, are generated from

the responses received by SS1. Where the nodes indicate locations that TP1 may have passed. Any graphs containing a node where TP1 was observed are associated as active, or candidate, paths, i.e. paths that TP1 may have traversed. With a depth first search, only SRs adjacent to a location where TP1 was observed are queried. SRs issuing a positive reply are treated as starting points for candidate graphs because they have observed TP1. Next, all SRs adjacent to those that responded with a positive reply are queried. The process of moving the query/response process out one hop at a time is referred to as a round. This process is repeated until all participating routers have been queried or all SRs in a round respond with a negative reply indicating that they have not observed TP1. When a negative reply is received, it is associated as inactive path data.

[0048] — When SS1 has determined an ingress point for TP1, it may send a message to IDS1 indicating that a solution has been found (step 420). Often it will be desirable to have the participating router closest to the ingress point close off the ingress path used by TP1. As such, SS1 may send a message to the respective participating router instructing it to close off the ingress path using known techniques (step 422). SS1 may also archive path solutions, data sent, data received, and the like either locally or remotely. Furthermore, SS1 may communicate information about source path isolation attempts to devices at remote locations coupled to a network. For example, SS1 may communicate information to a network operations center (NOC), a redundant source path isolation server, or to a data analysis facility for post processing.

[0049] — Here it is noted that as SS1 attempts to build a trace of the path taken by TP1, multiple paths may emerge as a result of hash collisions occurring in participating routers. When collisions occur, they act as false positives in the sense that SS1 interprets the collision as an indication that a desired TP1 has been observed. Fortunately the occurrences of hash collisions can be mitigated. One mechanism for reducing hash collisions is to compute large hash values over the packets since the chances of collisions rise as the number of bits comprising the hash value decreases. Another mechanism for reducing collisions is to control the density of the hash tables in the memories of participating routers. That is, rather than computing a single hash value and setting a single bit for an observed packet, a plurality of hash values are computed for each observed packet using several unique hash functions. This produces a corresponding number of unique hash values for each observed packet. While this approach fills the router's

hash table at a faster rate, the reduction in the number of hash collisions makes the tradeoff worthwhile in many instances.

Figure 5

EXEMPLARY DATA STRUCTURE FOR STORING TRACE INFORMATION

{0050}—Fig. 5 illustrates an exemplary data structure 500 stored in a database (not shown) in a memory on a source path isolation server. Data structure 500 stores information used in conjunction with performing source path isolation of a target packet. While Fig. 5 illustrates one data structure, it will be obvious to those skilled in the relevant arts that a plurality of data structures may be employed and that the data structures may include additional parameters and take on different forms from those of the exemplary data structure discussed herein.

{0051}—Data structure 500 is comprised of a record R(1) containing attributes, or parameters, having data associated therewith. In the upper left portion of Fig. 5 are three parameters associated with the entire record R(1), namely a target packet attribute, shown as Target ID, a time attribute, shown as Time, and a source attribute, shown as Source. These attributes together serve as a handle for R(1) to facilitate storage into, and recall from, a machine-readable memory (not shown). Here Target ID is associated with unique information associated with a particular target packet (TP) received from a detection device such as an IDS or firewall. Time may be used to identify either the time at which TP was received at an SS, the time that TP was received at a detection device, or the time that R(1) was opened. Source may be used to identify the link that TP was detected on by the detection device, or alternatively, source may be used to uniquely identify the detection device that forwarded TP to SS.

{0052}—Within 500 are exemplary column headings indicating still other attributes that may be used to facilitate source path isolation of TP. For example, a network component identification attribute, shown as node ID, may be used to identify particular nodes, such as routers, switches, bridges, or the like, within a network that have been queried by SS. Link may be used to identify the particular link on which TP was observed. A reply packet attribute, shown as Node Response, may be used to indicate if a queried node has observed TP. Node time may indicate the time, preferably using some common reference, at which a respective node

observed TP. Time is useful for assessing how long TP has been in the network and for performing comparisons with fields such as time to live (TTL). The attribute Transformed is used to track variants of TP in the event it has undergone a transformation. If TP has been transformed, it may be useful to have multiple entries associated the respective TP. For example in Fig. 5, node 04 has two entries for tracing an untransformed and a transformed version of TP. Status may be used to monitor network links associated with queried nodes. For example, a status of "ON" may indicate that a link is still active, i.e. carrying data traffic, while a status of "OFF" may indicate that a link has been disabled to exclude data traffic.

{0053} — Fig. 5 illustrates one exemplary embodiment of a data structure that may be used for facilitating source path isolation; however, variations of the data structure format and number of records may be readily employed without departing from the spirit of the invention. For example, the terms "YES/NO" and "ON/OFF" used in conjunction with node response, transformed, and status may be desirable when conveying information to an operator; however, flags such as 1 or 0 may also be used to indicate the status of various attributes. In addition, a plurality of records may be generated when performing source path isolation. Additionally, other column entries may be used in conjunction with, or in place of, those shown in Fig. 5. For example, it may be desirable to associate the hash value, or alternatively, the contents of TP with each record. It may also be desirable to have a record associated with each target packet encountered or, alternatively, with each detection device employed within a network. And, it may be desirable to have still other data structures or records associated with source path solutions that have been generated in response to detected TPs.

-Figure 6

One particularly troublesome type of attack is a self-replicating network-transferred computer program, such as a virus or worm, that is designed to annoy network users, deny network service by overloading the network, or damage target computers (e.g., by deleting files). A virus is a program that infects a computer or device by attaching itself to another program and propagating

itself when that program is executed, possibly destroying files or wiping out memory devices. A worm, on the other hand, is a program that can make copies of itself and spread itself through connected systems, using up resources in affected computers or causing other damage.

In recent years, viruses and worms have caused major network performance degradations and wasted millions of man-hours in clean-up operations in corporations and homes all over the world. Famous examples include the "Melissa" e-mail virus and the "Code Red" worm.

Various defenses, such as e-mail filters, anti-virus programs, and firewall mechanisms, have been employed against viruses and worms, but with limited success. The defenses often rely on computer-based recognition of known viruses and worms or block a specific instance of a propagation mechanism (i.e., block e-mail transfers of Visual Basic Script (.vbs) attachments). New viruses and worms have appeared, however, that evade existing defenses.

Accordingly, there is a need for new defenses to thwart the attack of known and yet-to-be-developed viruses and worms. There is also a need to trace the path taken by a virus or worm.

SUMMARY OF THE INVENTION

Systems and methods consistent with the present invention address these and other needs by providing a new defense that attacks malicious packets, such as viruses and worms, at their most common denominator (i.e., the need to transfer a copy of their code over a network to multiple target systems, where this code is generally the same for each copy, even though the rest of the message containing the virus or worm may vary). The systems and methods also provide the ability to trace the path of propagation back to the point of origin of the malicious packet (i.e., the place at which it was initially injected into the network).

In accordance with the principles of the invention as embodied and broadly described herein, a system detects the transmission of potentially malicious packets. The system receives packets

and generates hash values corresponding to each of the packets. The system may then compare the generated hash values to hash values corresponding to prior packets. The system may determine that one of the packets is a potentially malicious packet when the generated hash value corresponding to the one packet matches one of the hash values corresponding to one of the prior packets and the one prior packet was received within a predetermined amount of time of the one packet.

According to another implementation consistent with the present invention, a system for hampering transmission of a potentially malicious packet is disclosed. The system includes means for receiving a packet; means for generating one or more hash values from the packet; means for comparing the generated one or more hash values to hash values corresponding to prior packets; means for determining that the packet is a potentially malicious packet when the generated one or more hash values match one or more of the hash values corresponding to at least one of the prior packets and the at least one of the prior packets was received within a predetermined amount of time of the packet; and means for hampering transmission of the packet when the packet is determined to be a potentially malicious packet.

According to yet another implementation consistent with the present invention, a method for detecting a path taken by a potentially malicious packet is disclosed. The method includes storing hash values corresponding to received packets; receiving a message identifying a potentially malicious packet; generating hash values from the potentially malicious packet; comparing the generated hash values to the stored hash values; and determining that the potentially malicious packet was one of the received packets when one or more of the generated hash values match the stored hash values.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate the invention and, together with the description, explain the invention. In

the drawings.

FIG. 1 is a diagram of a system in which systems and methods consistent with the present invention may be implemented;

FIG. 2 is an exemplary diagram of a security server of FIG. 1 according to an implementation consistent with the principles of the invention;

FIG. 3 is an exemplary diagram of packet detection logic according to an implementation consistent with the principles of the invention;

FIGS. 4A and 4B illustrate two possible data structures stored within the hash memory of FIG. 3 in implementations consistent with the principles of the invention;

FIG. 5 is a flowchart of exemplary processing for detecting and/or preventing transmission of a malicious packet, such as a virus or worm, according to an implementation consistent with the principles of the invention;

FIG. 6 is a flowchart of exemplary processing for identifying the path taken through a network by a malicious packet, such as a virus or worm, according to an implementation consistent with the principles of the invention; and

FIG. 7 is a flowchart of exemplary processing for determining whether a malicious packet, such as a virus or worm, has been observed according to an implementation consistent with the principles of the invention.

DETAILED DESCRIPTION

The following detailed description of the invention refers to the accompanying drawings. The

same reference numbers in different drawings may identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims and equivalents.

Systems and methods consistent with the present invention provide mechanisms to detect and/or prevent the transmission of malicious packets and trace the propagation of the malicious packets through a network. Malicious packets, as used herein, may include viruses, worms, and other types of data with duplicated content, such as illegal mass e-mail (e.g., spam), that are repeatedly transmitted through a network.

According to implementations consistent with the present invention, the content of a packet may be hashed to trace the packet through a network. In other implementations, the header of a packet may be hashed. In yet other implementations, some combination of the content and the header of a packet may be hashed.

EXEMPLARY SYSTEM FOR PERFORMING METHOD

[0054] FIG. 6 illustrates a system 620 comprising a general-purpose computer that can be configured to practice disclosed embodiments. System 620 executes machine-readable code to perform the methods heretofore disclosed and includes a processor 602~~CONFIGURATION~~

FIG. 1 is a diagram of an exemplary system 100 in which systems and methods consistent with the present invention may be implemented. System 100 includes autonomous systems (ASs) 110-140 connected to public network (PN) 150. Connections made in system 100 may be via wired, wireless, and/or optical communication paths. While FIG. 1 shows four autonomous systems connected to a single public network, there can be more or fewer systems and networks in other implementations consistent with the principles of the invention.

Public network 150 may include a collection of network devices, such as routers (R1-R5) or switches, that transfer data between autonomous systems, such as autonomous systems 110-140.

In an implementation consistent with the present invention, public network 150 takes the form of the Internet, an intranet, a public telephone network, a wide area network (WAN), or the like.

An autonomous system is a network domain in which all network devices (e.g., routers) in the domain can exchange routing tables. Often, an autonomous system can take the form of a local area network (LAN), a WAN, a metropolitan area network (MAN), etc. An autonomous system may include computers or other types of communication devices (referred to as "hosts") that connect to public network 150 via an intruder detection system (IDS), a firewall, one or more border routers, or a combination of these devices.

Autonomous system 110, for example, includes hosts (H) 111-113 connected in a LAN configuration. Hosts 111-113 connect to public network 150 via an intruder detection system 114. Intruder detection system 114 may include a commercially-available device that uses rule-based algorithms to determine if a given pattern of network traffic is abnormal. The general premise used by an intruder detection system is that malicious network traffic will have a different pattern from normal, or legitimate, network traffic.

Using a rule set, intruder detection system 114 monitors inbound traffic to autonomous system 110. When a suspicious pattern or event is detected, intruder detection system 114 may take remedial action, or it can instruct a border router or firewall to modify operation to address the malicious traffic pattern. For example, remedial actions may include disabling the link carrying the malicious traffic, discarding packets coming from a particular source address, or discarding packets addressed to a particular destination.

Autonomous system 120 contains different devices from autonomous system 110. These devices aid autonomous system 120 in identifying and/or preventing the transmission of potentially malicious packets within autonomous system 120 and tracing the propagation of the potentially malicious packets through autonomous system 120 and, possibly, public network 150. While FIG. 1 shows only autonomous system 120 as containing these devices, other autonomous

systems, including autonomous system 110, may include them.

Autonomous system 120 includes hosts (H) 121-123, intruder detection system 124, and security server (SS) 125 connected to public network 150 via a collection of devices, such as security routers (SR11-SR14) 126-129. Hosts 121-123 may include computers or other types of communication devices connected, for example, in a LAN configuration. Intruder detection system 124 may be configured similar to intruder detection system 114.

Security server 125 may include a device, such as a general-purpose computer or a server, that performs source path identification when a malicious packet is detected by intruder detection system 124 or a security router 126-129. While security server 125 and intruder detection system 124 are shown as separate devices in FIG. 1, they can be combined into a single unit performing both intrusion detection and source path identification in other implementations consistent with the present invention.

FIG. 2 is an exemplary diagram of security sever 125 according to an implementation consistent with the principles of the invention. While one possible configuration of security server 125 is illustrated in FIG. 2, other configurations are possible.

Security server 125 may include a processor 202, main memory [[604]]204, read only memory (ROM) [[606]]206, storage device [[608]]208, bus [[610]]210, display [[612]]212, keyboard [[614]]214, cursor control [[616]]216, and communication interface [[618.]]218. Processor [[602]]202 may [[be]]include any type of conventional processing device that interprets and executes instructions.

Main memory [[604]]204 may [[be]]include a random access memory (RAM) or a similar type of dynamic storage device. Main memory 604 stores 204 may store information and instructions to be executed by processor [[602.]] 202. Main memory [[604]]204 may also be used for storing temporary variables or other intermediate information during execution of instructions by

processor [[602.]]202. ROM ~~606 stores~~206 may store static information and instructions for use by processor[[602.]]202. It will be appreciated that ROM [[606]]206 may be replaced with some other type of static storage device. Storage device[[608]]208, also referred to as a data storage device, may include any type of magnetic or optical media and their corresponding interfaces and operational hardware. Storage device ~~608 stores~~208 may store information and instructions for use by processor [[602.]] 202.

Bus 640~~includes~~210 may include a set of hardware lines (conductors, optical fibers, or the like) that allow for data transfer among the components of system [[620.]]security server 125. Display device [[612]]212 may be a cathode ray tube (CRT), liquid crystal display (LCD) or the like, for displaying information in an operator or machine-readable form. Keyboard [[614]]214 and cursor control [[616]]216 may allow the operator to interact with system [[620.]]security server 125. Cursor control [[616]]216 may ~~[[be]]~~include, for example, a mouse. In an alternative configuration, keyboard [[614]]214 and cursor control [[616]]216 can be replaced with a microphone and voice recognition ~~[[means]]~~mechanisms to enable an operator or machine to interact with system [[620.]]security server 125.

Communication interface [[618]]218 enables ~~system 620~~security server 125 to communicate with other devices/systems via any communications medium. For example, communication interface [[618]]218 may ~~[[be]]~~include a modem, an Ethernet interface to a LAN, an interface to the Internet, a printer interface, etc. Alternatively, communication interface [[618]]218 can ~~[[be]]~~include any other type of interface that enables communication between system 620~~security server 125~~ and other devices, systems, or networks. Communication interface [[618]]218 can be used in lieu of keyboard [[614]]214 and cursor control [[616]]216 to facilitate operator or machine remote control and communication with ~~system 620~~security server 125.

As will be described in detail below, ~~system 620~~security server 125 may ~~provide SS1 operating within AS1 with the ability to perform source path isolation~~provide SS1 operating within AS1 with the ability to perform source path isolation~~identification and/or prevention measures for a given TP.~~SS1-malicious packet that entered autonomous system 120. Security

~~server 125 may receive MPI from IDS1 and generate QM1 perform these functions in response to processor [[602]]202 executing sequences of instructions contained in, for example, memory [[604.]]204. Such instructions may be read into memory [[604]]204 from another computer-readable medium, such as storage device [[608]]208, or from another device coupled to bus [[610]]210 or coupled via communication interface 618. Execution of sequences of instructions contained in memory 604 causes processor 602 to perform the method described in conjunction with FIG. 4. For example, processor 602 may execute instructions to perform the functions of receiving a target packet (step 402), receiving replies from queried routers (step 408), and building a trace of the path traveled by TP (step 416). 218.~~

Alternatively, hard[[-]]wired circuitry may be used in place of or in combination with software instructions to implement the functions of SS1. ~~Thus, the disclosed embodiments of SS1 are not limited to any specific combination of hardware circuitry and software.~~ security server 125. For example, the functionality may be implemented in an application specific integrated circuit (ASIC), a field-programmable gate array (FPGA), or the like, either alone or in combination with other devices to provide desired functionality.

CONCLUSION

~~{0055} — As can be seen, the disclosed embodiments provide the functionality necessary to facilitate source path isolation of malicious packets in a network. While the preceding disclosure is directed to an Internet Protocol (IP) network, disclosed embodiments can be used in conjunction with other network protocols such as frame relay, asynchronous transfer mode (ATM), synchronous optical network (SONET), and the like. In addition, disclosed embodiments may be adapted to operate within different layers of a network such as the data link layer, network layer, transport layer or the like. Furthermore, the disclosed embodiments are not limited to particular network topologies or architectures.~~

~~{0056} — Furthermore the disclosed methods for implementing a source path isolation server (SS) are not limited to a single programming language or hardware architecture. For example, software for performing the functions of SS may be implemented in a high-level~~

programming language such as C, C++, LISP, or the like. Alternatively, software may be implemented in a lower level language such as assembly language, or a device specific language, where requirements such as speed must be met. Furthermore, SS may be configured to communicate with, and make information available to, other devices operatively connected to a network using known programming languages and techniques. For example, it may be desirable to have SS make source path isolation solutions available to an operator responsible for monitoring network security. In addition, SS can be implemented in a distributed fashion either by employing multiple processors or by having various components physically separated and coupled by a communication means such as a distributed bus, network, or the like. Also, it may be desirable to have SS communicate with one or more SRs over a dedicated network instead of using the network carrying data traffic among the SRs. For example, using a dedicated network may provide additional security, reliable bandwidth, or communication redundancy in the event that one or more links to an SR is disabled.

[0057] Query messages (QMs) and replies are not limited to a single network protocol or packet type. In many instances, it will be desirable to have QMs and replies transported using readily known protocols; however, customized protocols and message types can be used. For example, it may be desirable to employ a *smart packet* for sending QMs to participating routers. A smart packet is one that may contain a standard message, such as the data from a target packet, along with machine readable instructions for instructing a receiving device, such as an SR, to modify its operation in response to the contents of the executable instructions contained therein. Smart packets facilitate rapid responses to network intrusions by allowing an SR to modify operation soon after receiving a QM from a SS, or a forwarded QM from a participating router.

[0058] Furthermore, the disclosed methods can operate on encapsulated data such as would be encountered if network data were encrypted, converted from one network protocol to another, or a packet was split for transmission over more than one link. As can be seen, many variations of the disclosed embodiments are possible without departing from the spirit of the invention.

Therefore, the present embodiments are to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the

~~foregoing description, and all changes within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.~~

Returning to FIG. 1, security routers 126-129 may include network devices, such as routers, that may detect and/or prevent the transmission of malicious packets and perform source path identification functions. Security routers 127-129 may include border routers for autonomous system 120 because these routers include connections to public network 150. As a result, security routers 127-129 may include routing tables for routers outside autonomous system 120.

FIG. 3 is an exemplary diagram of packet detection logic 300 according to an implementation consistent with the principles of the invention. Packet detection logic 300 may be implemented within a device that taps one or more bidirectional links of a router, such as security routers 126-129. In another implementation, packet detection logic 300 may be implemented within a router, such as security routers 126-129. In the discussion that follows, it may be assumed that packet detection logic 300 is implemented within a security router.

Packet detection logic 300 may include hash processor 310 and hash memory 320. Hash processor 310 may include a conventional processor, an ASIC, a FPGA, or a combination of these that generates one or more representations of each received packet and records the packet representations in hash memory 320.

A packet representation will likely not be a copy of the entire packet, but rather it will include a portion of the packet or some unique value representative of the packet. Because modern routers can pass gigabits of data per second, storing complete packets is not practical because memories would have to be prohibitively large. By contrast, storing a value representative of the contents of a packet uses memory in a much more efficient manner. By way of example, if incoming packets range in size from 256 bits to 1000 bits, a fixed width number may be computed across fixed-sized blocks making up the content (or payload) of a packet in a manner that allows the entire packet to be identified. To further illustrate the use of representations, a 32-bit hash value,

or digest, may be computed across fixed-sized blocks of each packet. Then, the hash value may be stored in hash memory 320 or may be used as an index, or address, into hash memory 320. Using the hash value, or an index derived therefrom, results in efficient use of hash memory 320 while still allowing the content of each packet passing through packet detection logic 300 to be identified.

Systems and methods consistent with the present invention may use any storage scheme that records information about each packet in a space-efficient fashion, that can definitively determine if a packet has not been observed, and that can respond positively (i.e., in a predictable way) when a packet has been observed. Although systems and methods consistent with the present invention can use virtually any technique for deriving representations of packets, for brevity, the remaining discussion will use hash values as exemplary representations of packets having passed through a participating router.

Hash processor 310 may determine a hash value over successive, fixed-sized blocks in the payload field (i.e., the contents) of an observed packet. For example, hash processor 310 may hash each successive 64-byte block following the header field. As described in more detail below, hash processor 310 may use the hash results of the hash operation to recognize duplicate occurrences of packet content and raise a warning if it detects packets with replicated content within a short period of time. Hash processor 310 may also use the hash results for tracing the path of a malicious packet through the network.

The hash value may be determined by taking an input block of data, such as a 64-byte block of a packet, and processing it to obtain a numerical value that represents the given input data. Suitable hash functions are readily known in the art and will not be discussed in detail herein. Examples of hash functions include the Cyclic Redundancy Check (CRC) and Message Digest 5 (MD5).

The resulting hash value, also referred to as a message digest or hash digest, is a fixed length

value. The hash value serves as a signature for the data over which it was computed. For example, incoming packets could have fixed hash value(s) computed over their content.

The hash value essentially acts as a fingerprint identifying the input block of data over which it was computed. Unlike fingerprints, however, there is a chance that two very different pieces of data will hash to the same value, resulting in a hash collision. An acceptable hash function should provide a good distribution of values over a variety of data inputs in order to prevent these collisions. Because collisions occur when different input blocks result in the same hash value, an ambiguity may arise when attempting to associate a result with a particular input.

Hash processor 310 may store a representation of each packet it observes in hash memory 320. Hash processor 310 may store the actual hash values as the packet representations or it may use other techniques for minimizing storage requirements associated with retaining hash values and other information associated therewith. A technique for minimizing storage requirements may use a bit array or Bloom filters for storing hash values.

Rather than storing the actual hash value, which can typically be on the order of 32 bits or more in length, hash processor 310 may use the hash value as an index for addressing a bit array within hash memory 320. In other words, when hash processor 310 generates a hash value for a fixed-sized block of a packet, the hash value serves as the address location into the bit array. At the address corresponding to the hash value, one or more bits may be set at the respective location thus indicating that a particular hash value, and hence a particular data packet content, has been seen by hash processor 310. For example, using a 32-bit hash value provides on the order of 4.3 billion possible index values into the bit array. Storing one bit per fixed-sized block rather than storing the block itself, which can be 521 bits long, produces a compression factor of 1:512. While bit arrays are described by way of example, it will be obvious to those skilled in the relevant art, that other storage techniques may be employed with out departing from the spirit of the invention.

Over time, hash memory 320 may fill up and the possibility of overwriting an existing index value increases. The risk of overwriting an index value may be reduced if the bit array is periodically flushed to other storage media, such as a magnetic disk drive, optical media, solid state drive, or the like. Alternatively, the bit array may be slowly and incrementally erased. To facilitate this, a time-table may be established for flushing the bit array. If desired, the flushing cycle can be reduced by computing hash values only for a subset of the packets passing through the router. While this approach reduces the flushing cycle, it increases the possibility that a target packet may be missed (i.e., a hash value is not computed over a portion of it).

FIGS. 4A and 4B illustrate two possible data structures that may be stored within hash memory 320 in implementations consistent with the principles of the invention. As shown in FIG. 4A, hash memory 320 may include indicator fields 412 and counter fields 414 addressable by corresponding hash addresses 416. Hash addresses 416 may correspond to possible hash values generated by hash processor 310.

Indicator field 412 may store one or more bits that indicate whether a packet block with the corresponding hash value has been observed by hash processor 310. Counter field 412 may record the number of occurrences of packet blocks with the corresponding hash value. Counter field 412 may periodically decrement its count for flushing purposes.

As shown in FIG. 4B, hash memory 320 may store additional information relating to a packet. For example, hash memory 320 may include link identifier (ID) fields 422 and status fields 424. Link ID field 422 may store information regarding the particular link upon which the packet arrived at packet detection logic 300. Status field 424 may store information to aid in monitoring the status of packet detection logic 300 or the link identified by link ID field 422.

In an alternate implementation consistent with the principles of the invention, hash memory 320 may be preprogrammed to store hash values corresponding to known malicious packets, such as known viruses and worms. Hash memory 320 may store these hash values separately from the

hash values of observed packets. In this case, hash processor 310 may compare a hash value for a received packet to not only the hash values of previously observed packets, but also to hash values of known malicious packets.

In yet another implementation consistent with the principles of the invention, hash memory 320 may be preprogrammed to store source addresses of known sources of legitimate duplicated content, such as packets from a multicast server, a popular page on a web server, an output from a mailing list "exploder" server, or the like. In this case, hash processor 310 may compare the source address for a received packet to the source addresses of known sources of legitimate duplicated content.

EXEMPLARY PROCESSING FOR MALICIOUS PACKET DETECTION

FIG. 5 is a flowchart of exemplary processing for detecting and/or preventing transmission of a malicious packet, such as a virus or worm, according to an implementation consistent with the principles of the invention. The processing of FIG. 5 may be performed by packet detection logic 300 within a tap device, a security router, such as security router 126, or other devices configured to detect and/or prevent transmission of malicious packets. In other implementations, one or more of the described acts may be performed by other systems or devices within system 100.

Processing may begin when packet detection logic 300 receives, or otherwise observes, a packet (act 505). Hash processor 310 may generate one or more hash values by hashing successive, fixed-sized blocks from the packet's payload field (act 510). Hash processor 310 may use a conventional technique to perform the hashing operation.

Hash processor 310 may optionally compare the generated hash value(s) to hash values of known viruses and/or worms within hash memory 320 (act 515). In this case, hash memory 320 may be preprogrammed to store hash values corresponding to known viruses and/or worms. If

one or more of the generated hash values match one of the hash values of known viruses and/or worms, hash processor 310 may take remedial actions (acts 520 and 525). The remedial actions may include raising a warning for a human operator, delaying transmission of the packet, requiring human examination before transmission of the packet, dropping the packet and possibly other packets originating from the same Internet Protocol (IP) address as the packet, sending a Transmission Control Protocol (TCP) close message to the sender thereby preventing complete transmission of the packet, disconnecting the link on which the packet was received, and/or corrupting the packet content in a way likely to render any code contained therein inert (and likely to cause the receiver to drop the packet).

If the generated hash value(s) do not match any of the hash values of known viruses and/or worms, or if such a comparison was not performed, hash processor 310 may optionally determine whether the packet's source address indicates that the packet was sent from a legitimate source of duplicated packet content (i.e., a legitimate "replicator") (act 530). For example, hash processor 310 may maintain a list of legitimate replicators in hash memory 320 and check the source address of the packet with the addresses of legitimate replicators on the list. If the packet's source address matches the address of one of the legitimate replicators, then hash processor 310 may end processing of the packet. For example, processing may return to act 505 and await receipt of the next packet.

Otherwise, hash processor 310 may determine whether any prior packets with the same hash value(s) have been received (act 535). For example, hash processor 310 may use each of the generated hash value(s) as an address into hash memory 320. Hash processor 310 may then examine indicator field 412 (FIG. 4) at each address to determine whether the one or more bits stored therein indicate that a prior packet has been received.

If there were no prior packets received with the same hash value(s), then hash processor 310 may record the generated hash value(s) in hash memory 320 (act 540). For example, hash processor 310 may set the one or more bits stored in indicator field 412, corresponding to each of the

generated hash values, to indicate that the corresponding packet was observed by hash processor 310. Processing may then return to act 505 to await receipt of the next packet.

If hash processor 310 determines that a prior packet has been observed with the same hash value, hash processor 310 may determine whether the packet is potentially malicious (act 545). Hash processor 310 may use a set of rules to determine whether to identify a packet as potentially malicious. For example, the rules might specify that more than times (where times > 1) packets with the same hash value have to be observed by hash processor 310 before the packets are identified as potentially malicious. The rules might also specify that these packets have to have been observed by hash processor 310 within a specified period of time of one another. The reason for the latter rule is that, in the case of malicious packets, such as viruses and worms, multiple packets will likely pass through packet detection logic 300 within a short period of time.

A packet may contain multiple hash blocks that partially match hash blocks associated with prior packets. For example, a packet that includes multiple hash blocks may have somewhere between one and all of its hashed content blocks match hash blocks associated with prior packets. The rules might specify the number of blocks and/or the number and/or length of sequences of blocks that need to match before hash processor 310 identifies the packet as potentially malicious.

When hash processor 310 determines that the packet is not malicious (e.g., not a worm or virus), such as when less than x number of packets with the same hash value or less than a predetermined number of the packet blocks with the same hash values are observed or when the packets are observed outside the specified period of time, hash processor 310 may record the generated hash value(s) in hash memory 320 (act 540). For example, hash processor 310 may set the one or more bits stored in indicator field 412, corresponding to each of the generated hash values, to indicate that the corresponding packet was observed by hash processor 310. Processing may then return to act 505 to await receipt of the next packet.

When hash processor 310 determines that the packet may be malicious, then hash processor 310

may take remedial actions (act 550). In some cases, it may not be possible to determine whether the packet is actually malicious because there is some probability that there was a false match or a legitimate replication. As a result, hash processor 310 may determine the probability of the packet actually being malicious based on information gathered by hash processor 310.

The remedial actions may include raising a warning for a human operator, saving the packet for human analysis, dropping the packet, corrupting the packet content in a way likely to render any code contained therein inert (and likely to cause the receiver to drop the packet), delaying transmission of the packet, requiring human examination before transmission of the packet, dropping other packets originating from the same IP address as the packet, sending a TCP close message to the sender thereby preventing complete transmission of the packet, and/or disconnecting the link on which the packet was received. Some of the remedial actions, such as dropping or corrupting the packet, may be performed when the probability that the packet is malicious is above some threshold. This may greatly slow the spread rate of a virus or worm without completely stopping legitimate traffic that happened to match a suspect profile.

EXEMPLARY PROCESSING FOR SOURCE PATH IDENTIFICATION

FIG. 6 is a flowchart of exemplary processing for identifying the path taken through a network by a malicious packet, such as a virus or worm, according to an implementation consistent with the principles of the invention. The processing of FIG. 6 may be performed by a security server, such as security server 125, or other devices configured to trace the paths taken by malicious packets. In other implementations, one or more of the described acts may be performed by other systems or devices within system 100.

Processing may begin with intruder detection system 124 detecting a malicious packet. Intruder detection system 124 may use conventional techniques to detect the malicious packet. For example, intruder detection system 124 may use rule-based algorithms to identify a packet as part of an abnormal network traffic pattern. When a malicious packet is detected, intruder

detection system 124 may notify security server 125 that a malicious packet has been detected within autonomous system 120. The notification may include the malicious packet or portions thereof along with other information useful for security server 125 to begin source path identification. Examples of information that intruder detection system 124 may send to security server 125 along with the malicious packet include time-of-arrival information, encapsulation information, link information, and the like.

After receiving the malicious packet, security server 125 may generate a query that includes the malicious packet and any additional information desirable for facilitating communication with participating routers, such as security routers 126-129 (acts 605 and 610). Examples of additional information that may be included in the query are, but are not limited to, destination addresses for participating routers, passwords required for querying a router, encryption keying information, time-to-live (TTL) fields, information for reconfiguring routers, and the like. Security server 125 may then send the query to security router(s) located one hop away (act 615). The security router(s) may analyze the query to determine whether they have seen the malicious packet. To make this determination, the security router(s) may use processing similar to that described below with regard to FIG. 7.

After processing the query, the security router(s) may send a response to security server. The response may indicate that the security router has seen the malicious packet, or alternatively, that it has not. It is important to observe that the two answers are not equal in their degree of certainty. If a security router does not have a hash matching the malicious packet, the security router has definitively not seen the malicious packet. If the security router has a matching hash, however, then the security router has seen the malicious packet or a packet that has the same hash value as the malicious packet. When two different packets, having different contents, hash to the same value it is referred to as a hash collision.

The security router(s) may also forward the query to other routers or devices to which they are connected. For example, the security router(s) may forward the query to the security router(s)

that are located two hops away from security server, which may forward the query to security router(s) located three hops away, and so on. This forwarding may continue to include routers or devices within public network 150 if these routers or devices have been configured to participate in the tracing of the paths taken by malicious packets. This approach may be called an inward-out approach because the query travels a path that extends outward from security server 125. Alternatively, an outward-in approach may be used.

Security server 125 receives the responses from the security routers indicating whether the security routers have seen the malicious packet (acts 620 and 625). If a response indicates that the security router has seen the malicious packet, security server 125 associates the response and identification (ID) information for the respective security router with active path data (act 630). Alternatively, if the response indicates that the security router has not seen the malicious packet, security server 125 associates the response and the ID information for the security router with inactive path data (act 635).

Security server 125 uses the active and inactive path data to build a trace of the potential paths taken by the malicious packet as it traveled, or propagated, across the network (act 640). Security server 125 may continue to build the trace until it receives all the responses from the security routers (acts 640 and 645). Security server 125 may attempt to build a trace with each received response to determine the ingress point for the malicious packet. The ingress point may identify where the malicious packet entered autonomous system 120, public network 150, or another autonomous system.

As security server 125 attempts to build a trace of the path taken by the malicious packet, several paths may emerge as a result of hash collisions occurring in the participating routers. When hash collisions occur, they act as false positives in the sense that security server 125 interprets the collision as an indication that the malicious packet has been observed. Fortunately, the occurrences of hash collisions can be mitigated. One mechanism for reducing hash collisions is to compute large hash values over the packets since the chances of collisions rise as the number

of bits comprising the hash value decreases. Another mechanism to reduce false positives resulting from collisions is for each security router (e.g., security routers 126-129) to implement its own unique hash function. In this case, the same collision will not occur in other security routers.

A further mechanism for reducing collisions is to control the density of the hash tables in the memories of participating routers. That is, rather than computing a single hash value and setting a single bit for an observed packet, a plurality of hash values may be computed for each observed packet using several unique hash functions. This produces a corresponding number of unique hash values for each observed packet. While this approach fills the hash table at a faster rate, the reduction in the number of hash collisions makes the tradeoff worthwhile in many instances. For example, Bloom Filters may be used to compute multiple hash values over a given packet in order to reduce the number of collisions and, hence, enhance the accuracy of traced paths.

When security server 125 has determined an ingress point for the malicious packet, it may notify intruder detection system 124 that the ingress point for the malicious packet has been determined (act 650). Security server 125 may also take remedial actions (act 655). Often it will be desirable to have the participating router closest to the ingress point close off the ingress path used by the malicious packet. As such, security server 125 may send a message to the respective participating router instructing it to close off the ingress path using known techniques.

Security server 125 may also archive copies of solutions generated, data sent, data received, and the like either locally or remotely. Furthermore, security server 125 may communicate information about source path identification attempts to devices at remote locations coupled to a network. For example, security server 125 may communicate information to a network operations center, a redundant security server, or to a data analysis facility for post processing.

EXEMPLARY PROCESSING FOR DETERMINING WHETHER A MALICIOUS PACKET

HAS BEEN OBSERVED

FIG. 7 is a flowchart of exemplary processing for determining whether a malicious packet, such as a virus or worm, has been observed according to an implementation consistent with the principles of the invention. The processing of FIG. 7 may be performed by packet detection logic 300 implemented within a security router, such as security router 126, or by other devices configured to trace the paths taken by malicious packets. In other implementations, one or more of the described acts may be performed by other systems or devices within system 100.

Processing may begin when security router 126 receives a query from security server 125 (act 705). As described above, the query may include a TTL field. A TTL field may be employed because it provides an efficient mechanism for ensuring that a security router responds only to relevant, or timely, queries. In addition, employing TTL fields may reduce the amount of data traversing the network between security server 125 and participating routers because queries with expired TTL fields may be discarded.

If the query includes a TTL field, security router 126 may determine if the TTL field in the query has expired (act 710). If the TTL field has expired, security router 126 may discard the query (act 715). If the TTL field has not expired, security router 126 may hash the malicious packet contained within the query at each possible starting offset within a block (act 720). Security router 126 may generate multiple hash values because the code body of a virus or worm may appear at any arbitrary offset within the packet that carries it (e.g., each copy may have an e-mail header attached that differs in length for each copy).

Security router 126 may then determine whether any of the generated hash values match one of the recorded hash values in hash memory 320 (act 725). Security router 126 may use each of the generated hash values as an address into hash memory 320. At each of the addresses, security router 126 may determine whether indicator field 412 indicates that a prior packet with the same hash value has been observed. If none of the generated hash values match a hash value in hash

memory 320, security router 126 does not forward the query (act 730), but instead may send a negative response to security server 125 (act 735).

If one or more of the generated hash values match a hash value in hash memory 320, however, security router 126 may forward the query to all of its output ports excluding the output port in the direction from which the query was received (act 740). Security router 126 may also send a positive response to security server 125, indicating that the packet has been observed (act 745). The response may include the address of security router 126 and information about observed packets that have passed through security router 126.

CONCLUSION

Systems and methods consistent with the present invention provide mechanisms to detect and/or prevent transmission of malicious packets, such as viruses and worms, and trace the propagation of the malicious packets through a network.

The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention.

For example, systems and methods have been described with regard to network-level devices. In other implementations, the systems and methods described herein may be used with a stand-alone device at the input or output of a network link or at other protocol levels, such as in mail relay hosts (e.g., Simple Mail Transfer Protocol (SMTP) servers).

While series of acts have been described with regard to the flowcharts of FIGS. 5-7, the order of the acts may differ in other implementations consistent with the principles of the invention. In addition, non-dependent acts may be performed concurrently.

Further, certain portions of the invention have been described as "logic" that performs one or more functions. This logic may include hardware, such as an application specific integrated circuit or a field programmable gate array, software, or a combination of hardware and software.

No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used. The scope of the invention is defined by the claims and their equivalents.